ZB-2018 Series
User Manual

Warranty

All products manufactured by ICP DAS are under warranty regarding defective materials for a period of one year, beginning from the date of delivery to the original purchaser.

Warning

ICP DAS assumes no liability for any damage resulting from the use of this product. ICP DAS reserves the right to change this manual at any time without notice. The information furnished by ICP DAS is believed to be accurate and reliable. However, no responsibility is assumed by ICP DAS for its use, not for any infringements of patents or other rights of third parties resulting from its use.

Copyright

Copyright @ 2010 by ICP DAS Co., Ltd. All rights are reserved.

Trademark

The names used for identification only may be registered trademarks of their respective companies.

Table of Contents

1. Introduction 5

1.1. Pin Assignments 6

1.2. Specifications 7

1.3. Wire Connection 9

1.4. Quick Start 10

1.5. Switch Descriptions 11

1.6. Connection 14

1.7. INIT Mode 17

1.8. Analog Input Type and Data Format Table 18

1.9. Calibration 21

1.10. Technical Support 22

2. DCON Protocol 23

2.1. %AANNTTCCFF 27

2.2. #AA 28

2.3. #AAN 29

2.4. $AA0 31

2.5. $AA1 33

2.6. $AA2 35

2.7. $AA3 36

2.8. $AA5 37

2.9. $AA5VV 38

2.10. $AA6 40

2.11. $AA7CiRrr 42

2.12. $AA8Ci 44

2.13. $AA9 46

2.14. $AA9SNNNN 47

2.15. $AA9Ci 48

2.16. $AA9SNNNNCi 49
2.17. $AAA 50
2.18. $AAAi 51
2.19. $AAF 52
2.20. $AAM 53
2.21. $AAS1 54
2.22. ~** 55
2.23. ~AA0 56
2.24. ~AA1 58
2.25. ~AA2 60
2.26. ~AA3ETT 62
2.27. ~AAC 64
2.28. ~AACN 65
2.29. ~AAEV 66
2.30. ~AAO(Name) 68
2.31. @AACH 69
2.32. @AACHi 70
2.33. @AACHCi 71
2.34. @AACL 72
2.35. @AACLi 73
2.36. @AACLCi 74
2.37. @AADHCi 75
2.38. @AADI 76
2.39. @AADLCi 78
2.40. @AAHI(data)CiT 79
2.41. @AALO(data)CiT 81
2.42. @AARH 83
2.43. @AARHCi 84

2.44. @AARHi *85*

2.45. @AARL *86*

2.46. @AARLCi *87*

2.47. @AARLi *88*

3. Modbus RTU Protocol **89**

3.1. 02 (0x02) Read Input Status *90*

3.2. 04 (0x04) Read Input Channels *91*

3.3. 70 (0x46) Read/Write Module Settings *92*

3.3.1 Sub-function 00 (0x00) Read module name 93

3.3.2 Sub-function 07 (0x07) Read type code 94

3.3.3 Sub-function 08 (0x08) Set type code 95

3.3.4 Sub-function 32 (0x20) Read firmware version 96

3.3.5 Sub-function 37 (0x25) Read channel enabled/disabled status 97

3.3.6 Sub-function 38 (0x26) Set channel enable/disable 98

3.3.7 Sub-function 41 (0x29) Read miscellaneous settings 99

3.3.8 Sub-function 42 (0x2A) Write miscellaneous settings 100

3.3.9 Sub-function 43 (0x2B) Read the CJC offset 101

3.3.10 Sub-function 44 (0x2C) Write the CJC offset 102

3.3.11 Sub-function 45 (0x2D) Read the CJC enabled/disabled status 103

3.3.12 Sub-function 46 (0x2E) Enable/disable CJC 104

3.3.13 Sub-function 47 (0x2F) Read the CJC update setting 105

3.3.14 Sub-function 48 (0x30) Write the CJC update settings 106

3.4. Address Mappings *107*

3.5. Engineering Data Format Table *110*

4. Troubleshooting **112**

4.1. Communicating with the module *112*

4.2. Reading Data *113*

5. Appendix **114**

5.1. Dual Watchdog Operation *114*

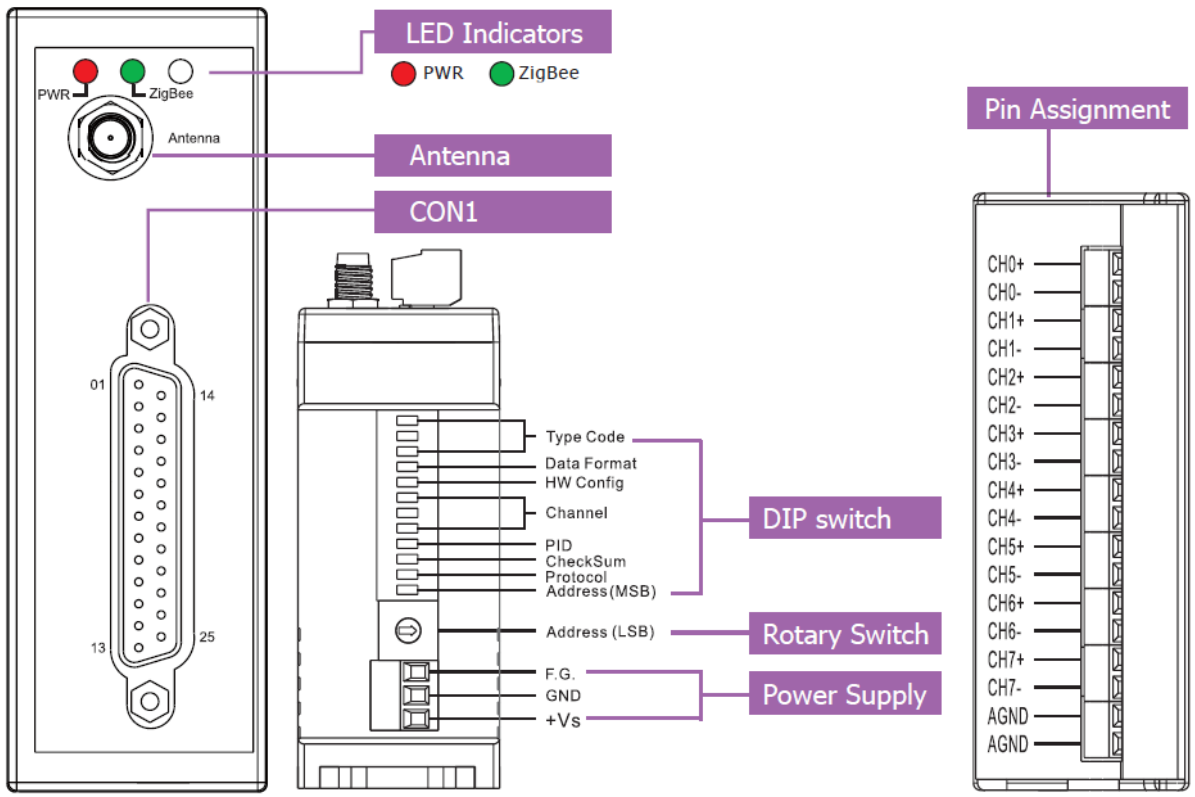
1.Introduction

The ZB-2000 series is a family of wireless ZigBee data acquisition modules that provide analog-to-digital, digital-to-analog, digital input/output, timer/counter and other functions. These modules can be remotely controlled using a set of DCON or Modbus RTU commands. The ZB-2018 series are 8-channel analog input modules that can be configured via hardware. Please refer to Section 1.5, “Switch Descriptions”, for details.

The ZB-2000 series modules are not able to work as standalone module and need to be connected to a ZigBee host device, such as the ZB-2550(P)(-T) or the ZB-2570(P)(-T), in order to communicate with the ZB-2000 series. Please refer to “ZigBee converter quick start“ at the following link:
http://ftp.icpdas.com/pub/cd/usbcd/napdos/zigbee/zigbee_converter/ for more information.

1.1. Pin Assignments

ZB-2018 Series + CN1824

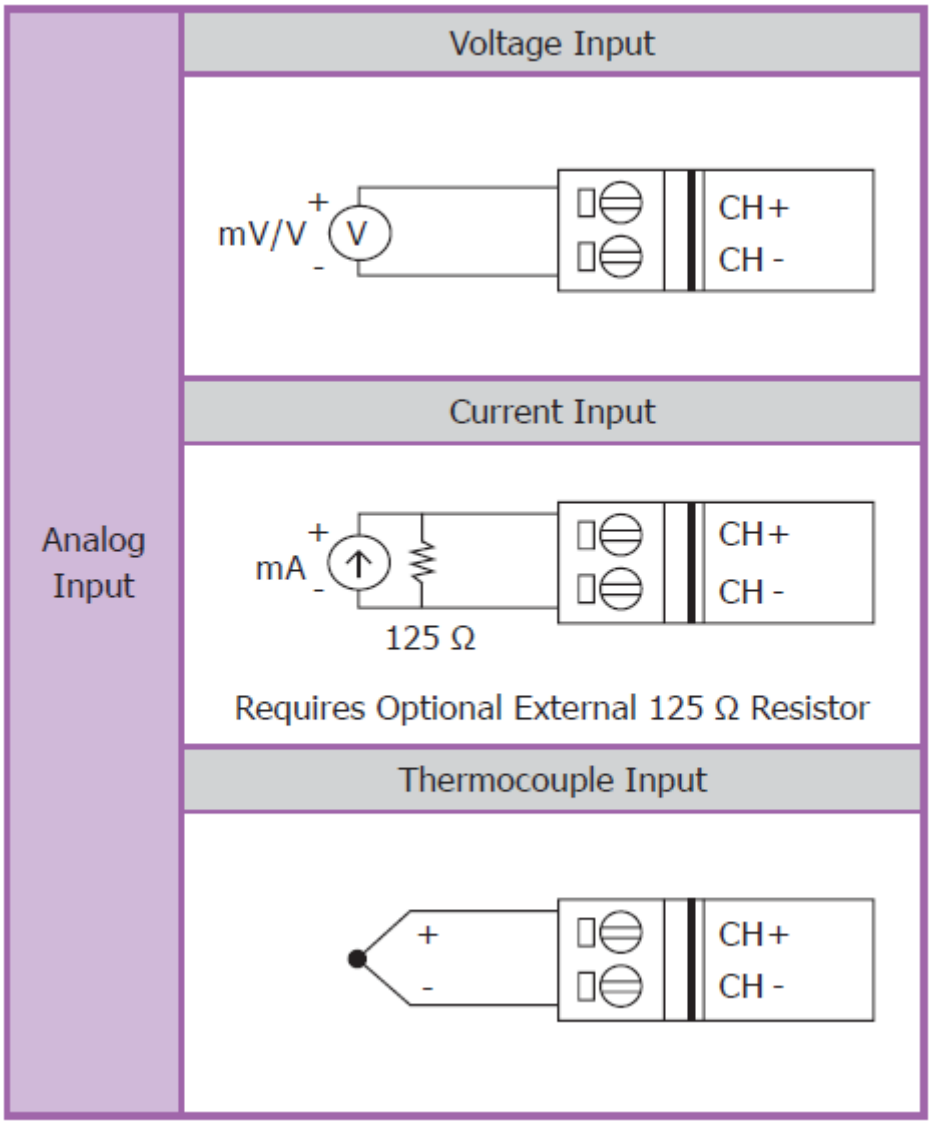


1.2. Specifications

Analog Input	
Input Channel	8 Differential
Input Type	+/-15 mV, +/-50 mV, +/-100 mV, +/-500 mV, +/-1V, +/-2.5V, +/-20 mA, 0 ~ 20 mA, 4 ~ 20 mA (Requires Optional External 125 Ω Resistor). Thermocouple (J, K, T, E. R. S, B, N, C, L, M, LDIN43710)
Resolution	16-bit
Sampling Rate	10 Samples/Sec. (Total)
Accuracy	+/-0.1% of FSR
-3dB Bandwidth	15.7 Hz
Zero Drift	+/-10 μV/°C
Span Drift	+/-25 ppm/°C
Common Mode Rejection	86 dB min.
Normal Mode Rejection	100 dB
Input Impedance	>400 kΩ
Open Thermocouple Detection	Yes
Overvoltage Protection	240 Vrms
Intra-module Isolated, Field to Logic	3000 V_{DC}
ESD Protection	+/-4 kV Contact for each channel
Communication Interface	
Wireless	ZigBee, IEEE 802.15.4 Standard
Antenna	2.4 GHz-3 dBi Omni-Directional antenna
Protocols	Supports DCON and Modbus RTU Protocols
Hot Swap	By Rotary and DIP switch
LED Indicators	
Power	1 LED, red
ZigBee Communication	1 LED, green
Power	
Power Consumption	0.88 W max.
Mechanical	

Flammability	Fire Retardant Materials (UL94-V0 Level)
Dimensions (W x L x H)	33 mm x 87 mm x 107 mm
Installation	DIN-Rail
Environment	
Operating Temperature	-25 °C ~ +75 °C
Storage Temperature	-30 °C ~ +80 °C
Humidity	10 ~ 90% RH, non-condensing
Certification	
CE	EN 301 489-1 V1.8.1(2008-04) EN 301 489-17 V2.1.1(2009-05) EN 55022:2006/A1:2007 EN 55024:1998/A1:2001/A2:2003
FCC	FCC Part 15 Subpart B Class A

1.3. Wire Connection

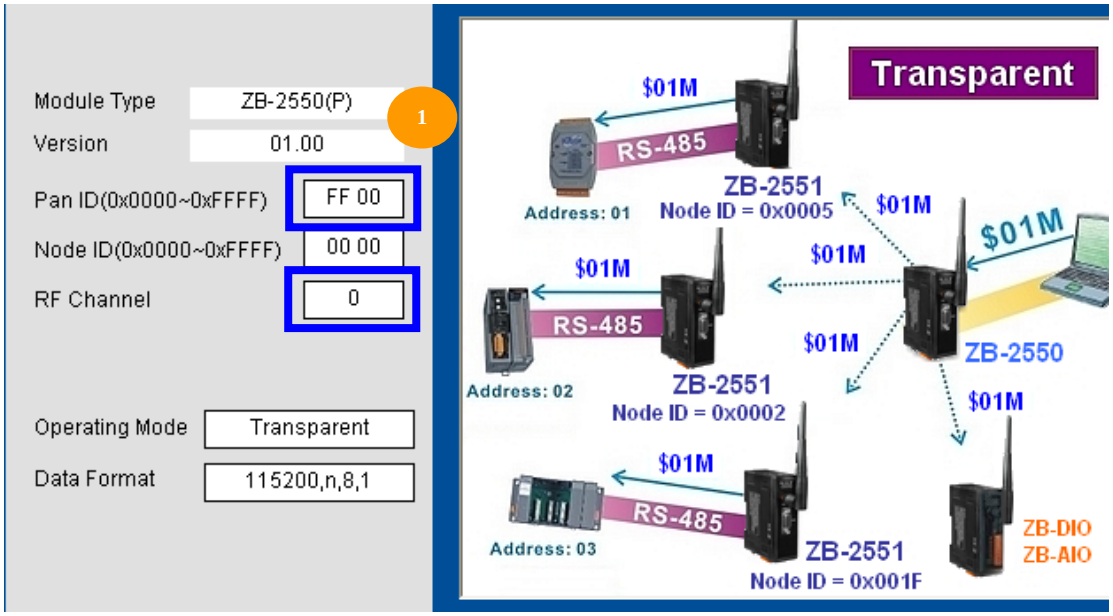


1.4. Quick Start

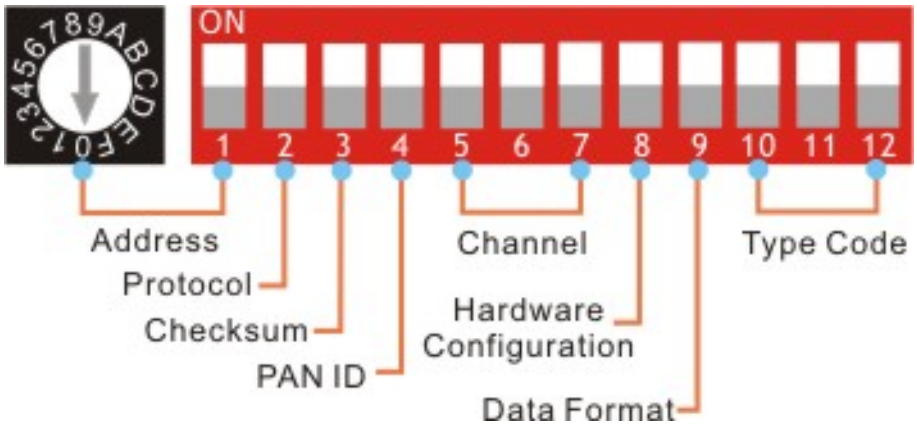
This Quick Start document describes the methods used to quickly set up and test ZB-2000 series modules using the ICP DAS DCON Utility.

First, you must set the ZB-2570(P)(-T)/ZB-2550(P)(-T) before using any ZB-2000 modules because the ZB-2570(P)(-T)/ZB-2550(P)(-T) is a ZigBee Net Server. For more information about the ZB-2570(P)(-T)/ZB-2550(P)(-T), please refer to the “ZigBee converter quick start“ at the following link:
http://ftp.icpdas.com/pub/cd/usbcd/napdos/zigbee/zigbee_converter

If you have already installed ZB-257x/ZB-255x Utility, you only need to set the “PAN ID” and the “ZB RF Channel” for the ZB-2570(P)(-T)/ZB-2550(P)(-T) to be same as the ZB-AIO setting. The ZB-AIO will then operate correctly.



1.5. Switch Descriptions



A descriptions of each dip switch is as follows:

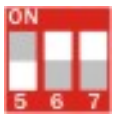
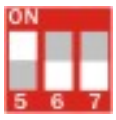
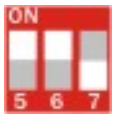
1. **Address:** The module address is defined using two parts. The first part is dip switch 1 and the second is a 16-position rotary switch. The address is a hexadecimal value that allows you set from 0x01 to 0x1F (0x00 is for initial mode). Dip switch 1 defines the high 4 bits of the address value and the 16-position rotary switch defines the low 4 bits of the address.
   The address value is equal to the ZigBee PAN ID value. A unique node ID should be set for all ZigBee slave devices, such as ZigBee IO modules, ZigBee converters and ZigBee repeaters.
   The ZB-2018 series will change to INIT mode when the address value is 0. Refer to Section 1.7 “INIT mode” for more information.

2. **Protocol:** Dip switch 2 defines the protocol. The ON position is the “Modbus RTU” protocol and OFF position is the “DCON” protocol.

3. **Checksum:** Dip switch 3 defines the checksum status. The ON positon enables the checksum and the OFF position disables the checksum. This option is only effective when the DCON protocol is enabled.

4. **PAN ID:** Dip switch 4 defines the ZigBee network PAN ID. Only 0xFF00 or 0xFF01 is allowed for the ZigBee IO series . 0xFF01 is selected by moving the dip switch to the ON position and 0xFF00 is selected by moving the dip switch to the OFF position.

5. **Channel:** Dip switches 5-7 define the ZigBee operating channel. The configuration is as follows:

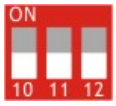
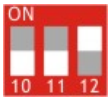
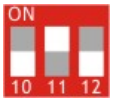
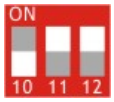
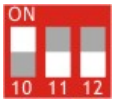
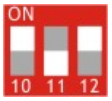
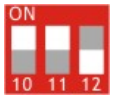
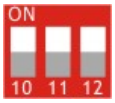
Switch value	Channel	Switch value	Channel	Switch value	Channel
	0		1		2
	3		4		9
	14		15		

6. **Hardware configuration:** Dip switch 8 defines whether the ZB-2018 series type code is configured using firmware or hardware. The ON position defines that configuration is via hardware and the OFF position defines that configuration is performed via firmware. The difference between firmware and hardware configuration are as follows:

Type Code is firmware configuration	Type Code is hardware configuration
Allows different type codes to be set for each channel.	All channels will use the same type code that is defined using thevalues set via dip switches 10-12.
The type code value is configured based on the EEPROM of the ZB-2018 series.	The type code value is configured based on the dip switch of the ZB-2018 series.
The data format is configured using software commands.	The data format is configured by the dip switch. Data format configuration commands are ignored in this mode.
When you use a command to read the current type code, you will get the value that is stored in the EEPROM of the ZB-2018 series.	When you use a command to read the current type code, you will get the value that has been configured via the dip switch of the ZB-2018series.

7. **Data Format:** Dip switch 9 defines the data format of the ZB-2018 series. The ON position defines hex format and OFF position defines engineering format. This dip switch is only valid when the “Hardware configuration” dip switch is in the ON position.

8. **Type Code:** Dip switches 10-12 define the input type code of the ZB-2018 series, sa shown below.

Switch value	Type code	Switch value	Type code	Switch value	Type code
	0x00		0x01		0x02
	0x03		0x04		0x05
	0x06		0x07		

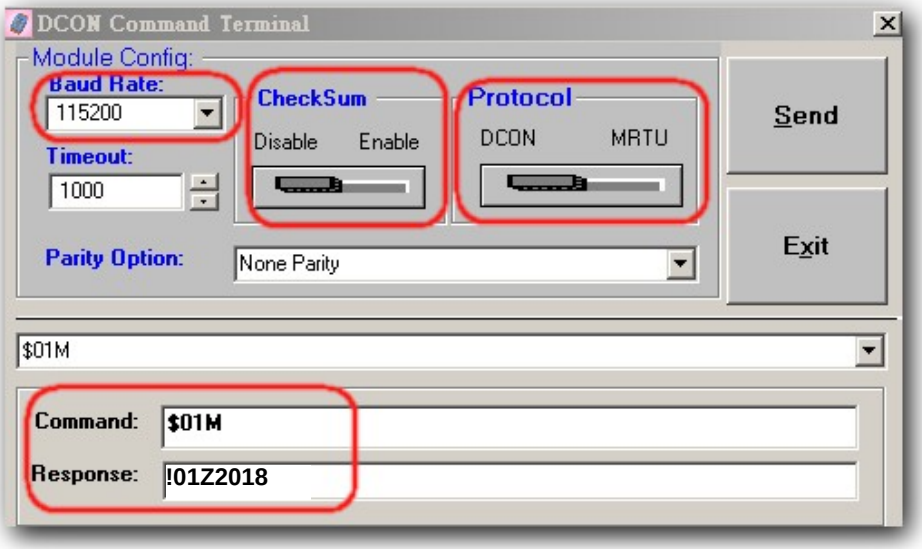
1.6. Connection

A ZigBee host must already exist in a ZigBee network. If you want to communicate with the ZB-2000 IO modules, you need to use a ZB-2550(P)(-T) or a ZP-2570(P) to connect to your controller.

To create a ZigBee network, the “Channel” and “PAN ID” parameters of all ZigBee modules should be individually configure to the same value.

For instance, use the software utility to configure the PAN ID of the ZB-2550(P)(-T) as 0xFF00 and the Channel as 0. Then set dip switches 4-7 of the ZB-2018 series to the OFF position to set the PAN ID to 0xFF00 and the Channel to 0. Based on the above configuration, the ZB-2550(P)(-T), the ZB-2018 series can now communicate with each other. Dip switches 2 and 3 to the OFF position to set the protocol for the module to DCON and disable the checksum.

Rotating the rotary switch of the ZB-2018 series to position 1 will set the module address to 0x01. You can then use the “DCON Utility” on the host PC that is connected to the ZB-2550(P)(-T) to send a command to the ZB-2018 series. The response you receive should be as below:



ZB-2018 series are command based data acquisition modules. A number of commands are provided that can be used to configure and read AI data. Refer to Section 2 for details.

The ZB-2018 series also support the Modbus RTU protocol. The Configuration command format for the Modbus RTU is as follows:

Field ❶	Field ❷	Field ❸	Field ❹ ~ Field *n	Field (❹+*n)
Module Address	Function code	Sub function	Configuration field	CRC16

*n: This value depends on the Sub-function code.

Eg: To set the channel 0, 1, 2 and 3 enable and channel 4, 5, 6 and 7 disable, the following command should be sent:

01 46 26 0F BA 69

The supported AI/O commands are as follows:

Function code	Description
0x01	Read coils
0x02	Read discrete inputs
0x03	Read multiple registers
0x04	Read multiple input registers
0x05	Write single coils
0x0F	Write multiple coils

Eg: To read the current AI value of channels 0 to 7, the following command should be sent:

01 03 00 00 00 07 04 08

Eg: To set the filter to 50Hz, the following command should be sent:

01 05 01 02 FF 00 2C 06

To install the ZigBee AIO module, follow the steps below:

1. Connect the analog input.
2. Connect the ZigBee AIO module to the power supply using the +Vs and GND terminals.
3. In order to read data from the input channels when using the DCON protocol, send either a #AA or #AAN command to the module. See Sections 2.2 and 2.3 for details. When using the Modbus RTU protocol, use the Function 04h to read the data from the input channels. See Section 3.2 for details.

1.7. INIT Mode

Each ZigBee module has an internal EEPROM that is used to store its configuration such as module address, ZigBee PAN ID, ZigBee channel etc. If you forget the module's configuration information, you can use INIT mode to reset the ZB-2018 series to the default settings, then you can re-configure the module. To change to INIT mode, you only need to adjust the address value to 0.

The default settings for ZB AIO modules are:

- Protocol: DCON
- Module Address: 0

1.8. Analog Input Type and Data Format Table

Type Code	Input Type	Data Format	+F.S.	-F.S.
00	-15 to +15 mA	Engineering unit	+15.000	-15.000
		% of FSR	+100.00	-100.00
		2's comp HEX	FFFF	8000
01	-50 to +50 mV	Engineering unit	+50.000	-50.000
		% of FSR	+100.00	-100.00
		2's comp HEX	7FFF	8000
02	-100 to +100 mV	Engineering unit	+100.00	-100.00
		% of FSR	+100.00	-100.00
		2's comp HEX	7FFF	8000
03	-500 to +500 mV	Engineering unit	+500.00	-500.00
		% of FSR	+100.00	-100.00
		2's comp HEX	7FFF	8000
04	-1 to +1 V	Engineering unit	+1.0000	-1.0000
		% of FSR	+100.00	-100.00
		2's comp HEX	7FFF	8000
05	-2.5 to +2.5 V	Engineering unit	+2.5000	-2.5000
		% of FSR	+100.00	-100.00
		2's comp HEX	7FFF	8000
06	-20 to +20 mA	Engineering unit	+20.000	-20.000
		% of FSR	+100.00	-100.00
		2's comp HEX	7FFF	8000
07	0 to +20 mA	Engineering unit	+20.000	+00.000
		% of FSR	+100.00	+000.00
		2's comp HEX	FFFF	0000
0E	Type J Thermocouple -210 ~ 760℃	Engineering unit	+760.00	-210.00
		% of FSR	+100.00	-027.63
		2's comp HEX	FFFF	DCA2
0F	Type K Thermocouple -270 ~ 1372℃	Engineering unit	+1372.0	-0270.0
		% of FSR	+100.00	-019.68
		2's comp HEX	7FFF	E6D0
10	Type T Thermocouple	Engineering unit	+400.00	-270.00
		% of FSR	+100.00	-067.50

	-270 ~ 400℃	2's comp HEX	7FFF	DCA2
11	Type E Thermocouple -270 ~ 1000℃	Engineering unit	+1000.0	-0270.0
		% of FSR	+100.00	-027.00
		2's comp HEX	7FFF	DD71
12	Type R Thermocouple 0 ~ 1768℃	Engineering unit	+1768.0	-0000.0
		% of FSR	+100.00	-000.00
		2's comp HEX	7FFF	0000
13	Type S Thermocouple 0 ~ 1768℃	Engineering unit	+1768.0	-0000.0
		% of FSR	+100.00	-000.00
		2's comp HEX	7FFF	0000
14	Type B Thermocouple 0 ~ 1820℃	Engineering unit	+1820.0	-0000.0
		% of FSR	+100.00	-000.00
		2's comp HEX	7FFF	0000
15	Type N Thermocouple -270 ~ 1300℃	Engineering unit	+1300.0	-0270.0
		% of FSR	+100.00	-020.77
		2's comp HEX	7FFF	E56B
16	Type C Thermocouple 0 ~ 2320℃	Engineering unit	+2320.0	-0000.0
		% of FSR	+100.00	-000.00
		2's comp HEX	7FFF	0000
17	Type L Thermocouple -200 ~ 800℃	Engineering unit	+800.00	-200.00
		% of FSR	+100.00	-025.00
		2's comp HEX	7FFF	E000
18	Type M Thermocouple -200 ~ 100℃	Engineering unit	+100.00	-200.00
		% of FSR	+050.00	-100.00
		2's comp HEX	4000	8000
19	Type $L_{DIN43710}$ Thermocouple -200 ~ 800℃	Engineering unit	+900.00	-200.00
		% of FSR	+100.00	-022.22
		2's comp HEX	7FFF	E38E
1A	4 to +20 mA	Engineering unit	+20.000	+04.000
		% of FSR	+100.00	+000.00
		2's comp HEX	FFFF	0000

Analog Inputy Over/Under Range Reading

	Over Range	Under Range
Engineering Unit	+9999.9	-9999.9
% of FSR	+999.99	-999.99
2's Complement HEX	7FFF	8000

Analog Input Over/Under Range Reading when using the Modbus RTU protocol

Over Range	Under Range
7FFFh	8000h

Data Format Setting (FF)

7	6	5	4	3	2	1	0
FS	Reserved	MS	Reserved			DF	

Key	Description
DF	Data format 00: Engineering units 01: % of FSR (Full Scale Range) 10: 2's complement hexadecimal
MS	Mode settings 0: Normal mode 1: Fast mode
FS	Filter setting 0: 60 Hz rejection 1: 50 Hz rejection.

1.9. Calibration

Warning: *Performing calibration is not recommended until the process is fully understood.*

The calibration procedure is as follows:

1. Warm up the module for at least 30 minutes.
2. Set the type code to the type you wish to calibrate. Refer to Sections 1.8 and 2.11 for details.
3. Enable calibration. Refer to Section 2.29 for details.
4. Apply the zero calibration voltage/current.
5. Send the zero calibration command. Refer to Section 2.5 for details.
6. Apply the span calibration voltage/current.
7. Send the span calibration command. Refer to Section 2.4 for details.
8. Repeat steps 3 to 7 three times.

Notes:

1. Connect the calibration voltage/current to channel 0.
2. Calibration voltages and currents are shown as below.
3. Switch to the DCON protocol mode before calibrating. Refer to Section 1.5 for details of the switching protocol.

Calibration voltage type used by the ZB-2018 series:

Type Code	00	01	02	03	04	05	06
Zero Input	0mV	0mV	0mV	0mV	0V	0V	0mA
Span Input	+15mV	+50mV	+100mV	+500mV	+1V	+2.5V	20mA

1.10. Technical Support

Should you encounter any problems while using the ZB-2018series module, and are unable to find the help you need in this manual or on our website, please contact ICP DAS Product Support.

Email: support@icpdas.com
Website: http://www.icpdas.com/service/support.htm

When requesting technical support, be prepared to provide the following information about your system:

1. Module name and serial number: The serial number can be found printed on the barcode label attached to the cover of the module.
2. Firmware version: See Sections 2.19 and 3.3.4 for information regarding the command used to identify the firmware version.
3. Host configuration (type and operating system)
4. If the problem is reproducible, please give full details describing the procedure used to reproduce the problem.
5. Any specific error messages displayed. If a dialog box with an error message is displayed, please include the full text of the dialog box, including the text in the title bar.
6. If the problem involves other programs or hardware devices, please describe the details of the problem in full.
7. Any comments and suggestions related to the problem are welcome.

ICP DAS will reply to your request by email within three business days.

2.DCON Protocol

All communication with ZB AIO modules consists of commands generated by the host and responses transmitted by the ZB AIO modules. Each module has a unique ID number that is used for addressing purposes and is stored in non-volatile memory. The ID is 01 by default and can be changed by transmitting the prescribed user command. All commands to the modules contain the ID address, meaning that only the addressed module will respond. The only exception to this is commands ~**(Section 2.22) which are sent to all modules, but in both of these cases, the modules do not reply to the command.

Command Format:

Leading Character	Module Address	Command	[CHKSUM]	CR

Response Format:

Leading Character	Module Address	Data	[CHKSUM]	CR

CHKSUM A 2-character checksum which is present when the checksum setting is enabled. See Section 1.8 (Data Format Settings) for details.

CR End of command character, carriage return (0x0D)

Checksum Calculation:

1. Calculate the ASCII code sum of all the characters in the command/response string, except for the carriage return character (CR).
2. The checksum is equal to the sum masked by 0FFh.

Example:

Command string: $012(CR)

1. Sum of the string = “$”+”0”+”1”+”2” = 24h+30h+31h+32h = B7h
2. Therefore the checksum is B7h, and so CHKSUM = “B7”
3. The command string with the checksum = $012B7(CR)

Response string: !01200600(CR)

1. Sum of the string = “!”+”0”+”1”+”2”+”0”+”0”+”6”+”0”+”0” = 21h+30h+31h+32h+30h+30h+36h+30h+30h = 1AAh
2. Therefore the checksum is AAh, and so CHKSUM = “AA”
3. The response string with the checksum = !01200600AA(CR)

Note:
All characters should be in upper case.

General Command Sets			
Command	**Response**	**Description**	**Section**
%AANNTTCCFF	!AA	Sets the module configuration	2.1
#AA	>(Data)	Reads data from the analog inputs	2.2
#AAN	>(Data)	Reads data from the analog input of a channel	2.3
$AA0	!AA	Performs a zero calibration	2.4
$AA1	!AA	Performs a span calibration	2.5
$AA2	!AANNTTCCFF	Reads the module configuration	2.6
$AA5	!AAS	Reads the module reset status	2.8
$AA5VV	!AA	Enables/Disables the channel	2.9
$AA6	!AAVV	Reads the enabled/disabled status of the channel	2.10
$AA7CiRrr	!AA	Sets the range configuration of a channel	2.11
$AA8Ci	!AACiRrr	Reads the range configuration of a channel	2.12
$AAF	!AA(Data)	Reads the firmware version	2.19
$AAM	!AA(Data)	Reads the module name	2.20
$AAS1	!AA	Reloads the default calibration parameters	2.21
~AAEV	!AA	Enables/Disables calibration	2.29
~AAO(Name)	!AA	Sets the module name	2.30
@AACH	!AA	Clears the high latches	2.31
@AACHi	!AA	Clears the high latch of a channel	2.32
@AACHCi	!AA	Clears the high latched alarm of a channel	2.33
@AACL	!AA	Clears lthe ow latches	2.34
@AACLi	!AA	Clears the low latch of a channel	2.35
@AACLCi	!AA	Clears the low latched alarm of a channel	2.36
@AADHCi	!AA	Disables the high alarm of a channel	2.37

@AADI	!AAHHLL	Reads the alarm status	2.38
@AADLCi	!AA	Disables the low alarm of a channel	2.39
@AAHI(data)CiT	!AA	Sets the the high alarm of a channel	2.40
@AALO(data)CiT	!AA	Sets the low alarm of a channel	2.41
@AARH	!AA(data)	Reads the high latches	2.42
@AARHCi	!AA(data)S	Reads the high alarm of a channel	2.43
@AARHi	!AA(data)	Reads the high latch of a channel	2.44
@AARL	!AA(data)	Reads the low latches	2.45
@AARLCi	!AA(data)S	Reads the low alarm of a channel	2.46
@AARLi	!AA(data)	Reads the low latch of a channel	2.47
CJC Command Sets			
Command	**Response**	**Description**	**Section**
$AA3	>(Data)	Reads the CJC	2.7
$AA9	!AA(Data)	Reads the CJC offset	2.13
$AA9SNNNN	!AA	Sets the CJC offset	2.14
$AA9Ci	!AA(Data)	Reads the CJC offset of a channel	2.15
$AA9SNNNNCi	!AA	Sets the CJC offset of a channel	2.16
$AAA	!AAi	Reads the CJC update setting	2.17
$AAAi	!AA	Sets the CJC update setting	2.18
~AAC	!AAN	Enables/Disables the CJC	2.27
~AACN	!AA	Reads the enabled/disabled status of the CJC	2.28
Host Watchdog Command Sets			
Command	**Response**	**Description**	**Section**
~**	No Response	Host is OK	2.22
~AA0	!AASS	Reads the Host Watchdog status	2.23
~AA1	!AA	Resets the Host Watchdog status	2.24
~AA2	!AAETT	Reads the Host Watchdog timeout settings	2.25
~AA3ETT	!AA	Sets the Host Watchdog timeout settings	2.26

2.1. %AANNTTCCFF

Description:
This command is used to set the configuration of a module.

Syntax:
%AANNTTCCFF[CHKSUM](CR)

%	Delimiter character
AA	The address of the module to be configured in hexadecimal format (00 to 1F)
NN	The address of the module to be configured in hexadecimal format (Same as AA)
TT	00 (Reserved)
CC	0A (Reserved)
FF	Used to set the data format, checksum, and filter settings (Section 1.8 for details).

Response:

Valid Command:	**!AA[CHKSUM](CR)**
Invalid Command:	**?AA[CHKSUM](CR)**

!	Delimiter for a valid command
?	Delimiter for an invalid command.
AA	Address of the module in hexadecimal format (00 to FF)

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

Examples:
Command: %0202000A80 Response: !02

Sets the data format of module 02 to 80 (50Hz rejection). The module returns a valid response.

Related Commands:
Section 2.6 $AA2

2.2. #AA

Description:
This command is used to read the data from all analog input channels

Syntax:
#AA[CHKSUM](CR)
Delimiter character
AA Address of the module to be read (00 to FF)

Response:
Valid Command: **>(Data)[CHKSUM](CR)**
Invalid Command: **?AA[CHKSUM](CR)**
> Delimiter character for a valid command
? Delimiter character for an invalid command
(Data) Data from all analog input channels, see Section 1.8 for the data format. Data from disabled channels is filled with space characters.
AA Address of the responding module (00 to FF)

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

Examples:
Command: #01 Response:
>+10.000+10.000+10.000+10.000+10.000+10
.000+10.000+10.000
Reads module 01 and receives the data in engineering format.

Related Commands:
Section 2.1 %AANNTTCCFF, Section 2.6 $AA2, Section 2.11 $AA7CiRrr

Related Topics:
Section 1.8 Configuration Tables.

2.3. #AAN

Description:
This command is used to read the analog input of the channel N.

Syntax:
#AAN[CHKSUM](CR)
Delimiter character
AA Address of the module to be read (00 to FF)
N The channel to be read, zero based.

Response:
Valid Command: **>(Data)[CHKSUM](CR)**
Invalid Command: **?AA[CHKSUM](CR)**
> Delimiter character for a valid command
? Delimiter character for an invalid command.
(An invalid command is returned if the specified channel is incorrect.)
(Data) Analog input data of the specified channel, see Section 1.8 for the data format. If the specified channel is disabled, then the data field will be filled with space characters.
AA Address of the responding module (00 to FF)

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

Examples:

Command: #032 Response: >+025.13

Reads data from channel 2 of module 03.

Command: #029 Response: ?02

Reads data from channel 9 of module 02. An error is returned because channel 9 is invalid.

Related Commands:

Section 2.1 %AANNTTCCFF, Section 2.6 $AA2

Related Topics:

Section 1.8 Configuration Tables.

2.4. $AA0

Description:
Performs a span calibration.

Syntax:
$AA0[CHKSUM](CR)
$ Delimiter character
AA Address of the module to be calibrated (00 to FF)
0 Command for the span calibration

Response:
Valid Command: **!AA[CHKSUM](CR)**
Invalid Command: **?AA[CHKSUM](CR)**
! Delimiter character for a valid command
? Delimiter character for an invalid command. An invalid command is returned if the specified channel is incorrect.
AA Address of the responding module (00 to FF)

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

Examples:
Command: $010 Response: !01
Performs a span calibration of module 01 and returns a valid response.
Command: $030 Response: ?03
Performs a span calibration of module 03. An invalid command is returned because the “enable calibration” command was not sent in advance.

Related Commands:
Section 2.5 $AA1, Section 2.29 ~AAEV

Related Topics:
Section 1.9 Calibration

Notes:
1. The “enable calibration” command, ~AAEV, and the “zero calibration” command, $AA1, must be sent before this command is used, see Sections 1.9 and 2.5 for details.

2.5. $AA1

Description:
Performs a zero calibration.

Syntax:
$AA1[CHKSUM](CR)
$ Delimiter character
AA Address of the module to be calibrated (00 to FF)
1 Command for the zero calibration

Response:
Valid Command: **!AA[CHKSUM](CR)**
Invalid Command: **?AA[CHKSUM](CR)**
! Delimiter character for a valid command
? Delimiter character for an invalid command. An invalid command is returned if the specified channel is incorrect.
AA Address of the responding module (00 to FF)

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

Examples:
Command: $011 Receive: !01
Performs a zero calibration of module 01 and returns a valid response.
Command: $031 Receive: ?03
Performs a zero calibration of module 03. An invalid command is returned because the “enable calibration” command was not sent in advance.

Related Commands:
Section 2.4 $AA0, Section 2.29 ~AAEV

Related Topics:
Section 1.9 Calibration

Notes:
1. The “enable calibration” command, ~AAEV, must be sent before this command is used, see Section 1.9 for details.
2. This command must be sent before the “span calibration” command, $AA1, is used.

2.6. $AA2

Description:
This command is used to read the configuration of a module.

Syntax:
$AA2[CHKSUM](CR)
$ Delimiter character
AA Address of the module to be read (00 to FF)
2 Command to read the module configuration

Response:
Valid Command: **!AATTCCFF[CHKSUM](CR)**
Invalid Command: **?AA[CHKSUM](CR)**
! Delimiter character for a valid command
? Delimiter character for an invalid command
AA Address of the responding module (00 to 1F)
TT 00 (Reserved).
CC 0A (Reserved).
FF Data format, checksum settings and filter settings of the module, see Section 1.8 for details.

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

Examples:
Command: $012 Response: !01000A00
Reads the configuration of module 01.
Command: $022 Response: !02000A20
Reads the configuration of module 02.

Related Commands:
Section 2.1 %AANNTTCCFF

Related Topics:
Section 1.8 Configuration Tables

2.7. $AA3

Description:
This command is used to read the CJC(cold junction compensation) temperature.

Syntax:
$AA3[CHKSUM](CR)
$ Delimiter character
AA Address of the module to be read (00 to FF)
3 Command to read the CJC temperature

Response:
Valid Command: **>(Data)[CHKSUM](CR)**
Invalid Command: **?AA[CHKSUM](CR)**
> Delimiter character for a valid command
? Delimiter character for an invalid command
AA Address of the responding module (00 to 1F)
(Data) CJC temperature in degrees Celsius, consisting of a sign byte, '+' or '-', and followed by 5 decimal digits with a fixed decimal point in tenths of a degree.

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

Examples:
Command: $013 Response: >+0031.2
Reads the CJC temperature of module 01 and returns with 31.2℃.

Related Commands:
Section 2.13 $AA9, Section 2.14 $AA9SNNNN, Section 2.17 $AAA, Section 2.18 $AAAi, Section 2.27 ~AAC, Section 2.28 ~AACN

2.8. $AA5

Description:
This command is used to read the reset status of a module.

Syntax:
$AA5[CHKSUM](CR)
$ Delimiter character
AA Address of the module to be read (00 to FF)
5 Command to read the module reset status

Response:
Valid Command: **!AAS[CHKSUM](CR)**
Invalid Command: **?AA[CHKSUM](CR)**
! Delimiter character for a valid command
? Delimiter character for an invalid command
AA Address of the responding module (00 to FF)
S Reset status of the module
1: This is the first time the command has been sent since the module was powered on.
0: This is not the first time the command has been sent since the module was powered on, which denotes that there has been no module reset since the last $AA5 command was sent.

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

Examples:
Command: $015 Response: !011
Reads the reset status of module 01. The response shows that it is a first time the $AA5 command has been sent since the module was powered-on.
Command: $015 Response: !010
Reads the reset status of module 01. The response shows that there has been no module reset since the last $AA5 command was sent.

2.9. $AA5VV

Description:
This command is used to specify the channels to be enabled.

Syntax:
$AA5VV[CHKSUM](CR)

$	Delimiter character
AA	Address of the module to be set (00 to FF)
5	Command to set the channels to enabled
VV	A two-digit hexadecimal value, where bit 0 corresponds to channel 0, bit 1 corresponds to channel 1, etc. When the bit is 1, it denotes that the channel is enabled, and 0 denotes that the channel is disabled.

Response:

Valid Command:	**!AA[CHKSUM](CR)**
Invalid Command:	**?AA[CHKSUM](CR)**

!	Delimiter character for a valid command
?	Delimiter character for an invalid command. An invalid command is returned if an attempt is made to enable a channel that is not present.
AA	Address of the responding module (00 to FF)

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

Examples:

Command: $0153A Response: !01

Enables channels 1, 3, 4, and 5 and disables all other channels on module 01. The module returns a valid response.

Command: $016 Response: !013A

Reads the channel status of module 01. The module returns a response of 3A, which denotes that channels 1, 3, 4, and 5 are enabled and all other channels are disabled.

Related Commands:

Section 2.10 $AA6

2.10. $AA6

Description:
This command is used to read the enabled/disabled status of each channel.

Syntax:
$AA6[CHKSUM](CR)

$	Delimiter character
AA	Address of the module to be read (00 to FF)
6	Command to read the channel status

Response:

Valid Command:	**!AAVV[CHKSUM](CR)**
Invalid Command:	**?AA[CHKSUM](CR)**

!	Delimiter character for a valid command
?	Delimiter character for an invalid command
AA	Address of the responding module (00 to FF)
VV	A two-digit hexadecimal value, where bit 0 corresponds to channel 0, bit 1 corresponds to channel 1, etc. When the bit is 1, it denotes that the channel is enabled, and 0 denotes that the channel is disabled.

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

Examples:

Command: $0153A Response: !01

Enables channels 1, 3, 4, and 5 and disables all other channels on module 01. The module returns a valid response.

Command: $016 Response: !013A

Reads the channel status of module 01 and returns a response of 3A, which denotes that channels 1, 3, 4, and 5 are enabled and all other channels are disabled.

Related Commands:

Section 2.9 $AA5VV

2.11. $AA7CiRrr

Description:
This command is used to set the type code of a channel.

Syntax:
$AA7CiRrr[CHKSUM](CR)

$	Delimiter character
AA	Address of the module to be set (00 to FF)
7	Command to set the channel range code
Ci	i specifies the input channel to be set (0-7)
Rrr	rr represents the type code of the channel to be set. Refer to the Temperature Sensor Type Setting table in Section 1.8.

Response:

Valid command:	**!AA [CHKSUM](CR)**
Invalid command:	**?AA[CHKSUM](CR)**

!	Delimiter character for a valid command
?	Delimiter character for an invalid command or invalid type code
AA	Address of the responding module (00 to FF)

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

Examples:

Command: $017C0R20 Response: !01

Sets the type code for channel 0 of module 01 to be 20 (PT100, -100 ~ +100℃) and the module returns a valid response.

Command: $027C5R28 Response: !02

Sets the type code for channel 5 of module 02 to 28 (Ni120, -80 ~ +100℃) and the module returns a valid response.

Command: $037C1R30 Response: ?03

Sets the type code for channel 1 of module 03 to 30. The module returns an invalid response because the type code is invalid.

Related Commands:

Section 2.12 $AA8Ci

Related Topics:

Section 1.8 Configuration Tables

2.12. $AA8Ci

Description:
This command is used to read the type code information for a channel.

Syntax:
$AA8Ci[CHKSUM](CR)

$	Delimiter character
AA	Address of the module to be read (00 to FF)
8	Command to read the type code of a channel
Ci	Specifies which channel to access for the type code information (0-7)

Response:
Valid command: **!AACiRrr[CHKSUM](CR)**
Invalid command: **?AA[CHKSUM](CR)**

!	Delimiter character for a valid command
?	Delimiter character for an invalid command or invalid channel
AA	Address of the responding module (00 to FF)
Ci	Specifies which input channel the type code information relates to.
Rrr	Represents the type code of the specified input channel. Refer to the Temperature Sensor Type Setting table in Section 1.8.

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

Examples:

Command: $018C0 Response: !01C0R20

Reads the input range of channel 0 of module 01 and returns 20 (PT100, -100 ~ +100℃).

Related Commands:

Section 2.11 $AA7CiRrr

Related Topics:

Section 1.8 Configuration Tables

2.13. $AA9

Description:
This command is used to read the CJC(cold junction compensation) offset value that is set by the $AA9SNNNN command(Section 2.14).

Syntax:
$AA9[CHKSUM](CR)
$ Delimiter character
AA Address of the module to be read (00 to FF)
9 Command to read the CJC offset value

Response:
Valid command: **!AA(Data)[CHKSUM](CR)**
Invalid command: **?AA[CHKSUM](CR)**
! Delimiter character for a valid command
? Delimiter character for an invalid command
AA Address of the responding module (00 to FF)
(Data) CJC offset value consisting of a sign byte, '+' or '-', followed by 4 hexadecimal digits. Each count is equal to 0.01℃.

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

Examples:
Command: $019 Response: !01+0010
Reads the CJC offset of module 01 and returns with +10 which means +0.16℃.

Related Commands:
Section 2.7 $AA3, Section 2.14 $AA9SNNNN, Section 2.17 $AAA, Section 2.18 $AAAi, Section 2.28 ~AACN

2.14. $AA9SNNNN

Description:
This command is used to set the CJC(cold junction compensation) offset value to adjust the error produced by the CJC sensor.

Syntax:
$AA9SNNNN[CHKSUM](CR)

$	Delimiter character
AA	Address of the module to be read (00 to FF)
9	Command to set the CJC offset value
S	Sign by, '+' or '-', of the offset value
NNNN	The absolute value of the offset in four hexadecimal digits, which must be less than or equal to 10000h. Each count is equal to 0.01℃.

Response:
Valid command: **!AA [CHKSUM](CR)**
Invalid command: **?AA[CHKSUM](CR)**

!	Delimiter character for a valid command
?	Delimiter character for an invalid command
AA	Address of the responding module (00 to FF)

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

Examples:
Command: $019+0010 Response: !01
Sets the CJC offset of module 01 to +0.16℃ and returns a valid response.

Related Commands:
Section 2.7 $AA3, Section 2.13 $AA9, Section 2.17 $AAA, Section 2.18 $AAAi, Section 2.28 ~AACN

2.15. $AA9Ci

Description:
This command is used to read the CJC(cold junction compensation) offset value of a channel.

Syntax:
$AA9Ci[CHKSUM](CR)

$	Delimiter character
AA	Address of the module to be read (00 to FF)
9	Command to read the CJC offset value of a channel
Ci	The channel to be clear, zero based

Response:
Valid command: **!AA(Data)[CHKSUM](CR)**
Invalid command: **?AA[CHKSUM](CR)**

!	Delimiter character for a valid command
?	Delimiter character for an invalid command
AA	Address of the responding module (00 to FF)
(Data)	CJC offset value consisting of a sign byte, '+' or '-', followed by 4 hexadecimal digits. Each count is equal to 0.01℃.

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

Examples:
Command: $019C0 Response: !01+0010
Reads the CJC offset of the channel 0 and returns with +10 which means +0.16℃.

Related Commands:
Section 2.7 $AA3, Section 2.16 $AA9SNNNNCi, Section 2.17 $AAA, Section 2.18 $AAAi, Section 2.28 ~AACN

2.16. $AA9SNNNNCi

Description:
This command is used to set the CJC(cold junction compensation) offset value of a channel.

Syntax:
$AA9SNNNNCi[CHKSUM](CR)

$	Delimiter character
AA	Address of the module to be read (00 to FF)
9	Command to set the CJC offset value
S	Sign by, '+' or '-', of the offset value
NNNN	The absolute value of the offset in four hexadecimal digits, which must be less than or equal to 10000h. Each count is equal to 0.01℃.
Ci	The channel to be clear, zero based

Response:
Valid command: **!AA [CHKSUM](CR)**
Invalid command: **?AA[CHKSUM](CR)**

!	Delimiter character for a valid command
?	Delimiter character for an invalid command
AA	Address of the responding module (00 to FF)

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

Examples:
Command: $019+0010C0 Response: !01
Sets the CJC offset of the channel 0 to +0.16℃ and returns a valid response.

Related Commands:
Section 2.7 $AA3, Section 2.15 $AA9Ci, Section 2.17 $AAA, Section 2.18 $AAAi, Section 2.28 ~AACN

2.17. $AAA

Description:
This command is used to read the CJC temperature update setting.

Syntax:
$AAA[CHKSUM](CR)
$ Delimiter character
AA Address of the module to be read (00 to FF)
A Command to read the CJC temperature update setting

Response:
Valid command: **!AAi[CHKSUM](CR)**
Invalid command: **?AA[CHKSUM](CR)**
! Delimiter character for a valid command
? Delimiter character for an invalid command
AA Address of the responding module (00 to FF)
i CJC temperature update setting
0 = CJC temperature update has been stopped
1 = CJC temperature update has been started
2 = update CJC temperature once only

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

Examples:
Command: $01A Response: !011
Reads the CJC temperature update setting of module 01 and returns with 1, meaning that the CJC temperature update has been started.

Related Commands:
Section 2.7 $AA3, Section 2.13 $AA9, Section 2.14 $AA9SNNNN, Section 2.18 $AAAi, Section 2.28 ~AACN

2.18. $AAAi

Description:
This command is used to set the CJC temperature update setting.

Syntax:
$AAAi[CHKSUM](CR)
$ Delimiter character
AA Address of the module to be read (00 to FF)
A Command to set the CJC temperature update setting
i CJC temperature update setting
0 = stop the CJC temperature update
1 = start the CJC temperature update(default value)
2 = update CJC temperature once only, after the command is received

Response:
Valid command: **!AA [CHKSUM](CR)**
Invalid command: **?AA[CHKSUM](CR)**
! Delimiter character for a valid command
? Delimiter character for an invalid command
AA Address of the responding module (00 to FF)

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

Examples:
Command: $01A1 Response: !01
Sets to start the CJC temperature update of module 01 and returns a valid response..

Related Commands:
Section 2.7 $AA3, Section 2.13 $AA9, Section 2.14 $AA9SNNNN, Section 2.17 $AAA, Section 2.28 ~AACN

2.19. $AAF

Description:
This command is used to read the firmware version of a module.

Syntax:
$AAF[CHKSUM](CR)
$ Delimiter character
AA Address of the module to be read (00 to FF)
F Command to read the firmware version

Response:
Valid command: **!AA(Data)[CHKSUM](CR)**
Invalid command: **?AA[CHKSUM](CR)**
! Delimiter character for a valid command
? Delimiter character for an invalid command
AA Address of the responding module (00 to FF)
(Data) Firmware version of the module in string format

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

Examples:
Command: $01F Response: !01A2.0
Reads the firmware version of module 01, and shows that it is version A2.0.
Command: $02F Response: !02B1.1
Reads the firmware version of module 02, and shows that it is version B1.1.

2.20. $AAM

Description:
This command is used to read the name of a module.

Syntax:
$AAM[CHKSUM](CR)
$ Delimiter character
AA Address of the module to be read (00 to FF)
M Command to read the module name

Response:
Valid command: **!AA(Data)[CHKSUM](CR)**
Invalid command: **?AA[CHKSUM](CR)**
! Delimiter character for a valid command
? Delimiter character for an invalid command
AA Address of the responding module (00 to FF)
(Data) Name string of the module in string format

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

Examples:
Command: $01M Response: !01Z2018
Reads the name of module 01 and returns the name “Z2018”.

Related Commands:
Section 2.30 ~AAO (Name)

2.21. $AAS1

Description:
This command is used to reload the factory default calibration parameters, including the internal calibration parameters.

Syntax:
$AAS1[CHKSUM](CR)

$	Delimiter character
AA	Address of the module where the default parameters are to be reloaded (00 to FF)
S1	Command to reload the factory default calibration parameters

Response:

Valid command:	**!AA[CHKSUM](CR)**
Invalid command:	**?AA[CHKSUM](CR)**
!	Delimiter character for a valid command
?	Delimiter character for an invalid command
AA	Address of the responding module (00 to FF)

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

Examples:
Command: $01S1 Response: !01

Sends a command to reload the factory default calibration parameters for module 01 and returns a valid response.

Related Topics:
Section 1.9 Calibration

2.22. ~**

Description:
This command is used to inform all modules that the host is OK.

Syntax:
~[CHKSUM](CR)**
~ Delimiter character
****** Host OK command

Response:
No response.

Examples:
Command: ~** No response
Sends a “Host OK” command to all modules.

Related Commands:
Section 2.23 ~AA0, Section 2.24 ~AA1, Section 2.25 ~AA2, Section 2.26 ~AA3ETT

Related Topics:
Section 5.1 Dual Watchdog Operation**.**

2.23. ~AA0

Description:
Reads the Host Watchdog status of a module.

Syntax:
~AA0[CHKSUM](CR)
~ Delimiter character
AA Address of the module to be read (00 to FF)
0 Command to read the Host Watchdog status

Response:
Valid command: **!AASS[CHKSUM](CR)**
Invalid command: **?AA[CHKSUM](CR)**
! Delimiter character for a valid command
? Delimiter character for an invalid command
AA Address of the responding module (00 to FF)
SS Two hexadecimal digits that represent the host watchdog status, where:
Bit 7: 0 indicates that the host watchdog is disabled, and 1 indicates that the host watchdog is enabled,
Bit 2: 1 indicates that a host watchdog timeout has occurred, and 0 indicates that no host watchdog timeout has occurred.
The host watchdog status is stored in EEPROM and can only be reset using the ~AA1 command.

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

Examples:

Command: ~010 Response: !0100

Reads the host watchdog status of module 01 and returns 00, meaning that the host watchdog is disabled and no host watchdog timeout has occurred.

Command: ~020 Response: !0204

Reads the host watchdog status of module 02 and returns 04, meaning that a host watchdog timeout has occurred.

Related Commands:

Section 2.22 ~**, Section 2.24 ~AA1, Section 2.25 ~AA2, Section 2.26 ~AA3ETT

Related Topics:

Section 5.1 Dual Watchdog Operation

2.24. ~AA1

Description:
This command is used to reset the host watchdog timeout status of a module.

Syntax:
~AA1[CHKSUM](CR)

~	Delimiter character
AA	Address of the module to be reset (00 to FF)
1	Command to reset the host watchdog timeout status

Response:

Valid command:	**!AA[CHKSUM](CR)**
Invalid command:	**?AA[CHKSUM](CR)**

!	Delimiter character for a valid command
?	Delimiter character for an invalid command
AA	Address of the responding module (00 to FF)

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

Examples:

Command: ~010 Response: !0104

Reads the host watchdog status of module 01 and shows that a host watchdog timeout has occurred.

Command: ~011 Response: !01

Resets the host watchdog timeout status of module 01 and returns a valid response.

Command: ~010 Response: !0100

Reads the host watchdog status of module 01 and shows that no host watchdog timeout has occurred.

Related Commands:

Section 2.22 ~**, Section 2.23 ~AA0, Section 2.25 ~AA2, Section 2.26 ~AA3ETT

Related Topics:

Section 5.1 Dual Watchdog Operation

2.25. ~AA2

Description:
This command is used to read the host watchdog timeout value of a module.

Syntax:
~AA2[CHKSUM](CR)

~	Delimiter character
AA	Address of the module to be read (00 to FF)
2	Command to read the host watchdog timeout value

Response:

Valid command : **!AAEVV[CHKSUM](CR)**
Invalid command: **?AA[CHKSUM](CR)**

!	Delimiter character for a valid command
?	Delimiter character for an invalid command
AA	Address of the responding module (00 to FF)
E	1: the host watchdog is enabled 0: the host watchdog is disabled
VV	Two hexadecimal digits to represent the timeout value in tenths of a second, for example, 01 denotes 0.1 seconds and FF denotes 25.5 seconds.

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

Examples:

Command: ~012 Response: !011FF

Reads the host watchdog timeout value of module 01 and returns FF, which denotes that the host watchdog is enabled and the host watchdog timeout value is 25.5 seconds.

Related Commands:

Section 2.22 ~**, Section 2.23 ~AA0, Section 2.24 ~AA1, Section 2.26 ~AA3ETT

Related Topics:

Section 5.1 Dual Watchdog Operation

2.26. ~AA3ETT

Description:
Enable/disable the host watchdog and sets the host watchdog timeout value of a module.

Syntax:
~AA3EVV[CHKSUM](CR)

~	Delimiter character
AA	Address of the module to be set (00 to FF)
3	Command to set the host watchdog
E	1: enables the host watchdog 0: disables the host watchdog
TT	Two hexadecimal digits to represent the timeout value in tenths of a second, for example, 01 denotes 0.1 seconds and FF denotes 25.5 seconds.

Response:

Valid command:	**!AA[CHKSUM](CR)**
Invalid command:	**?AA[CHKSUM](CR)**

!	Delimiter character for a valid command
?	Delimiter character for an invalid command
AA	Address of the responding module (00 to FF)

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

Examples:

Command: ~013164 Response: !01

Enables the host watchdog of module 01 and sets the host watchdog timeout value to 10.0 seconds. The module returns a valid response.

Command: ~012 Response: !01164

Reads the host watchdog timeout value of module 01. The module returns 164, which denotes that the host watchdog is enabled and the host watchdog timeout value is 10.0 seconds.

Related Commands:

Section 2.22 ~**, Section 2.23 ~AA0, Section 2.24 ~AA1, Section 2.25 ~AA2

Related Topics:

Section 5.1 Dual Watchdog Operation

Notes:

When a host watchdog timeout occurs, the host watchdog is disabled. The ~AA3ETT command should be sent again to re-enable the host watchdog.

2.27. ~AAC

Description:
This command is used to read the CJC(cold junction compensation) enabled/disabled status.

Syntax:
~AAC[CHKSUM](CR)
~ Delimiter character
AA Address of the module to be set (00 to FF)
C Command to read the CJC enabled/disabled status

Response:
Valid command: **!AAN[CHKSUM](CR)**
Invalid command: **?AA[CHKSUM](CR)**
! Delimiter character for a valid command
? Delimiter character for an invalid command
AA Address of the responding module (00 to FF)
N 0: CJC disabled
1: CJC enabled

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

Examples:
Command: ~01C Response: !011
Reads the CJC enabled/disabled status of module 01 and returns the CJC is enabled.

Related Commands:
Section 2.7 $AA3, Section 2.13 $AA9, Section 2.14 $AA9SNNNN, Section 2.17 $AAA, Section 2.18 $AAAi, Section 2.28 ~AACN

2.28. ~AACN

Description:
This command is used to enable/disable the CJC(cold junction compensation).

Syntax:
~AACN[CHKSUM](CR)

~	Delimiter character
AA	Address of the module to be set (00 to FF)
C	Command to enable/disable the CJC
N	0: CJC disabled 1: CJC enabled

Response:

Valid command:	**!AA[CHKSUM](CR)**
Invalid command:	**?AA[CHKSUM](CR)**

!	Delimiter character for a valid command
?	Delimiter character for an invalid command
AA	Address of the responding module (00 to FF)

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

Examples:
Command: ~01C0 Response: !01
Disables CJC of module 01 and returns a valid response.

Related Commands:
Section 2.7 $AA3, Section 2.13 $AA9, Section 2.14 $AA9SNNNN, Section 2.17 $AAA, Section 2.18 $AAAi, Section 2.27 ~AAC

2.29. ~AAEV

Description:
Enable/Disable the module calibration.

Syntax:
~AAEV[CHKSUM](CR)

~	Delimiter character
AA	Address of the module to be enabled/disabled (00 to FF)
E	Command to enable/disable calibration
V	1: enable calibration 0: disable calibration

Response:

Valid Command:	**!AA[CHKSUM](CR)**
Invalid Command:	**?AA[CHKSUM](CR)**
!	Delimiter character for a valid command
?	Delimiter character for an invalid command
AA	Address of the responding module (00 to FF)

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

Examples:

Command: $010 Response: ?01

Sends a command to perform a span calibration on module 01. An invalid response is returned because the “enable calibration” command has not yet been sent.

Command: ~01E1 Response: !01

Enables calibration on module 01 and returns a valid response.

Command: $010 Response: !01

Sends the command to perform a span calibration on module 01 and returns a valid response.

Related Commands:

Section 2.4 $AA0, Section 2.5 $AA1

Related Topics:

Section 1.9 Calibration

2.30. ~AAO(Name)

Description:
This command is used to set the name of a module.

Syntax:
~AAO(Name)[CHKSUM](CR)
~ Delimiter character
AA Address of the module to be set (00 to FF)
O Command to set the module name
(Name) New name of the module (max. 6 characters).

Response:
Valid command: **!AA[CHKSUM](CR)**
Invalid command: **?AA[CHKSUM](CR)**
! Delimiter character for a valid command
? Delimiter character for an invalid command
AA Address of the responding module (00 to FF)

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

Examples:
Command: ~01OZ2018 Response: !01
Sets the name of module 01 to be “Z2018” and returns a valid response.
Command: $01M Response: !01Z2018
Reads the name of module 01 and returns the name “Z2018”.

Related Commands:
Section 2.20 $AAM

2.31. @AACH

Description:
This command is used to clear the high latches.

Syntax:
@AACH [CHKSUM](CR)
@ Delimiter character
AA Address of the module to be set (00 to FF)
CH Command to clear the high latches

Response:
Valid command: **!AA[CHKSUM](CR)**
Invalid command: **?AA[CHKSUM](CR)**
! Delimiter character for a valid command
? Delimiter character for an invalid command
AA Address of the responding module (00 to FF)

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

Examples:
Command: @01RH0 Response: !01+05.000
Reads the high latch of channel 0 and returns +05.000.
Command: @01CH Response: !01
Clears the high latch of channel 0 and returns a valid response.
Command: @01RH0 Response: !01+00.000
Reads the high latch of channel 0 and returns +00.000.

Related Commands:
Section 2.32 @AACHi, Section 2.42 @AARH, Section 2.44 @AARHi

2.32. @AACHi

Description:
This command is used to clear the high latches of a channel.

Syntax:
@AACHi [CHKSUM](CR)
@ Delimiter character
AA Address of the module to be set (00 to FF)
CH Command to clear the high latches
i The channel to be clear, zero based

Response:
Valid command: **!AA[CHKSUM](CR)**
Invalid command: **?AA[CHKSUM](CR)**
! Delimiter character for a valid command
? Delimiter character for an invalid command
AA Address of the responding module (00 to FF)

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

Examples:
Command: @01RH1 Response: !01+06.000
Reads the high latch of channel 0 and returns +06.000.
Command: @01CH1 Response: !01
Clears the high latch of channel 0 and returns a valid response.
Command: @01RH1 Response: !01+00.000
Reads the high latch of channel 0 and returns +00.000.

Related Commands:
Section 2.31 @AACH, Section 2.42 @AARH, Section 2.44 @AARHi

2.33. @AACHCi

Description:
This command is used to clear the high alarm of a channel.

Syntax:
@AACHCi [CHKSUM](CR)
@ Delimiter character
AA Address of the module to be set (00 to FF)
CHC Command to clear the high alarm
i The channel to be clear, zero based

Response:
Valid command: **!AA[CHKSUM](CR)**
Invalid command: **?AA[CHKSUM](CR)**
! Delimiter character for a valid command
? Delimiter character for an invalid command
AA Address of the responding module (00 to FF)

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

Examples:
Command: @01DI Response: !018000
Reads the alarm status and returns high alarm of channel 7 has occurred.
Command: @01CHC7 Response: !01
Clears the high alarm of channel 7.
Command: @01DI Response: !010000
Reads the alarm status and returns high alarm and low alarm do not occurre.

Related Commands:
Section 2.31 @AACH, Section 2.42 @AARH, Section 2.44 @AARHi

2.34. @AACL

Description:
This command is used to clear the low latches.

Syntax:
@AACL [CHKSUM](CR)
@ Delimiter character
AA Address of the module to be set (00 to FF)
CL Command to clear the low latches

Response:
Valid command: **!AA[CHKSUM](CR)**
Invalid command: **?AA[CHKSUM](CR)**
! Delimiter character for a valid command
? Delimiter character for an invalid command
AA Address of the responding module (00 to FF)

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

Examples:
Command: @01RL0 Response: !01-05.000
Reads the low latch of channel 0 and returns -05.000.
Command: @01CL Response: !01
Clears the low latch of channel 0 and returns a valid response.
Command: @01RL0 Response: !01+00.000
Reads the low latch of channel 0 and returns +00.000.

Related Commands:
Section 2.35 @AACLi, Section 2.45 @AARL, Section 2.47 @AARLi

2.35. @AACLi

Description:
This command is used to clear the low latches of a channel.

Syntax:
@AACLi [CHKSUM](CR)
@ Delimiter character
AA Address of the module to be set (00 to FF)
CL Command to clear the low latches
i The channel to be clear, zero based

Response:
Valid command: **!AA[CHKSUM](CR)**
Invalid command: **?AA[CHKSUM](CR)**
! Delimiter character for a valid command
? Delimiter character for an invalid command
AA Address of the responding module (00 to FF)

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

Examples:
Command: @01RL1 Response: !01-06.000
Reads the low latch of channel 0 and returns -06.000.
Command: @01CL1 Response: !01
Clears the low latch of channel 0 and returns a valid response.
Command: @01RL1 Response: !01+00.000
Reads the low latch of channel 0 and returns +00.000.

Related Commands:
Section 2.34 @AACL, Section 2.45 @AARL, Section 2.47 @AARLi

2.36. @AACLCi

Description:
This command is used to clear the low alarm of a channel.

Syntax:
@AACLCi [CHKSUM](CR)
@ Delimiter character
AA Address of the module to be set (00 to FF)
CL Command to clear the low alarm
Ci The channel to be clear, zero based

Response:
Valid command: **!AA[CHKSUM](CR)**
Invalid command: **?AA[CHKSUM](CR)**
! Delimiter character for a valid command
? Delimiter character for an invalid command
AA Address of the responding module (00 to FF)

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

Examples:
Command: @01DI Response: !010080
Reads the alarm status and returns low alarm of channel 7 has occurred.
Command: @01CHC7 Response: !01
Clears the low alarm of channel 7.
Command: @01DI Response: !010000
Reads the alarm status and returns high alarms and low alarms do not occurre.

Related Commands:
Section 2.34 @AACL, Section 2.45 @AARL, Section 2.47 @AARLi

2.37. @AADHCi

Description:
This command is used to disable high alarm of a channel.

Syntax:
@AADHCi [CHKSUM](CR)
@ Delimiter character
AA Address of the module to be set (00 to FF)
DH Command to disable the high alarm
Ci The channel to be clear, zero based

Response:
Valid command: **!AA[CHKSUM](CR)**
Invalid command: **?AA[CHKSUM](CR)**
! Delimiter character for a valid command
? Delimiter character for an invalid command
AA Address of the responding module (00 to FF)

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

Examples:
Command: @01DHC0 Response: !01
Disables the high alarm of channel 0.
Command: @01DI Response: !01FEFF
Reads the alarm status and returns the high alarm of channel 0 is disabled and others are enabled.

Related Commands:
Section 2.38 @AADI

2.38. @AADI

Description:
This command is used to read the alarm status.

Syntax:
@AADI [CHKSUM](CR)
@ Delimiter character
AA Address of the module to be set (00 to FF)
DI Command to read the alarm status

Response:
Valid command: **!AAHHLL[CHKSUM](CR)**
Invalid command: **?AA[CHKSUM](CR)**
! Delimiter character for a valid command
? Delimiter character for an invalid command
AA Address of the responding module (00 to FF)
HH A two-digit hexadecimal value, where bit 0 corresponds to channel 0, bit 1 corresponds to channel 1, etc. When the bit is 1, it denotes that the alarm high of channel has occurred, and 0 denotes that the alarm high of channel has not occurred.
LL A two-digit hexadecimal value, where bit 0 corresponds to channel 0, bit 1 corresponds to channel 1, etc. When the bit is 1, it denotes that the alarm low of channel has occurred, and 0 denotes that the alarm low of channel has not occurred.

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

Examples:

Command: @01DI Response: !014008

Reads the alarm status and returns the high alarm of channel 6 and low alarm of channel 3 have ocurred.

Related Commands:

Section 2.37 @AADHCi, Section 2.39 @AADLCi, Section 2.40 @AAHI(data)CiT, Section 2.41 @AALO(data)CiT

2.39. @AADLCi

Description:
This command is used to disable low alarm of a channel.

Syntax:
@AADLCi [CHKSUM](CR)
@ Delimiter character
AA Address of the module to be set (00 to FF)
DLC Command to disable the low alarm
i The channel to be clear, zero based

Response:
Valid command: **!AA[CHKSUM](CR)**
Invalid command: **?AA[CHKSUM](CR)**
! Delimiter character for a valid command
? Delimiter character for an invalid command
AA Address of the responding module (00 to FF)

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

Examples:
Command: @01DLC5 Response: !01
Disables the low alarm of channel 0.
Command: @01DI Response: !01FFDF
Reads the alarm status and returns the low alarm of channel 5 is disabled and others are enabled.

Related Commands:
Section 2.38 @AADI

2.40. @AAHI(data)CiT

Description:
This command is used to set the high alarm of a channel.

Syntax:
@AAHI(data)CiT [CHKSUM](CR)

@	Delimiter character
AA	Address of the module to be set (00 to FF)
HI	Command to set the high alarm
(data)	High alarm limit, which consists with the data format. Refer to Section 1.8 for details.
Ci	The channel to be clear, zero based
T	Alarm type: M: momentary alarm L: latched alarm

Response:

Valid command:	**!AA[CHKSUM](CR)**
Invalid command:	**?AA[CHKSUM](CR)**

!	Delimiter character for a valid command
?	Delimiter character for an invalid command
AA	Address of the responding module (00 to FF)

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

Examples:

Command: @01HI+09.000C0M Response: !01

Sets the high alarm of channel 0. The high alarm limit is +09.000 and the type is momentary and returns a valid response.

Command: @01DI Response: !010100

Reads the alarm status and returns the high alarm of channel 0 is enabled and others are disabled.

Related Commands:

Section 2.38 @AADI

2.41. @AALO(data)CiT

Description:
This command is used to set the low alarm of a channel.

Syntax:
@AALO(data)CiT [CHKSUM](CR)

@	Delimiter character
AA	Address of the module to be set (00 to FF)
LO	Command to set the low alarm
(data)	High alarm limit, which consists with the data format. Refer to Section 1.8 for details.
Ci	The channel to be clear, zero based
T	Alarm type: M: momentary alarm L: latched alarm

Response:

Valid command:	**!AA[CHKSUM](CR)**
Invalid command:	**?AA[CHKSUM](CR)**
!	Delimiter character for a valid command
?	Delimiter character for an invalid command
AA	Address of the responding module (00 to FF)

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

Examples:

Command: @01LO-03.000C1L Response: !01

Sets the high alarm of channel 1. The low alarm limit is -03.000 and the type is latched and returns a valid response.

Command: @01DI Response: !010002

Reads the alarm status and returns the high alarm of channel 1 is enabled and others are disabled.

Related Commands:

Section 2.38 @AADI

2.42. @AARH

Description:
This command is used to read the high latches.

Syntax:
@AARH [CHKSUM](CR)
@ Delimiter character
AA Address of the module to be set (00 to FF)
RH Command to read the high latches

Response:
Valid command: **!AA(data)[CHKSUM](CR)**
Invalid command: **?AA[CHKSUM](CR)**
! Delimiter character for a valid command
? Delimiter character for an invalid command
AA Address of the responding module (00 to FF)
(data) High latches of all channels, see Section 1.8 for the data format.

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

Examples:
Command: @01RH Response:
!01+08.000+00.000+00.000+00.000+00.000+
00.000+00.000+00.000

Reads the high latches of module 01 and receives the data in engineering format.

Related Commands:
Section 2.31 @AACH, Section 2.32 @AACHi, Section 2.44 @AARHi

2.43. @AARHCi

Description:
This command is used to read the high alarm of a channel.

Syntax:
@AARHCi [CHKSUM](CR)
@ Delimiter character
AA Address of the module to be set (00 to FF)
RH Command to read the high alarm
Ci The channel to be clear, zero based

Response:
Valid command: **!AA(data)S[CHKSUM](CR)**
Invalid command: **?AA[CHKSUM](CR)**
! Delimiter character for a valid command
? Delimiter character for an invalid command
AA Address of the responding module (00 to FF)
(data) High latch of a channel, see Section 1.8 for the data format.
S Alarm type:
0: alarm disable
1: momentary alarm
2: latched alarm

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

Examples:
Command: @01RHC0 Response: !01+08.0002
Reads the high alarm of channel 0 and returns the high alarm limit is +08.000 and the type is latched.

Related Commands:
Section 2.40 @AAHI(data)CiT, Section 2.41 @AADHCi, Section 2.38 @AADI

2.44. @AARHi

Description:
This command is used to read the high latch of a channel.

Syntax:
@AARHi [CHKSUM](CR)
@ Delimiter character
AA Address of the module to be set (00 to FF)
RH Command to read the high latch
i The channel to be clear, zero based

Response:
Valid command: **!AA(data)[CHKSUM](CR)**
Invalid command: **?AA[CHKSUM](CR)**
! Delimiter character for a valid command
? Delimiter character for an invalid command
AA Address of the responding module (00 to FF)
(data) High latch of a channel, see Section 1.8 for the data format.

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

Examples:
Command: @01RH0 Response: !01+08.000
Reads the high latch of channel 0 and returns the data in engineering format.

Related Commands:
Section 2.31 @AACH, Section 2.32 @AACHi, Section 2.42 @AARH

2.45. @AARL

Description:
This command is used to read the low latches.

Syntax:
@AARL [CHKSUM](CR)
@ Delimiter character
AA Address of the module to be set (00 to FF)
RL Command to read the low latches

Response:
Valid command: **!AA(data)[CHKSUM](CR)**
Invalid command: **?AA[CHKSUM](CR)**
! Delimiter character for a valid command
? Delimiter character for an invalid command
AA Address of the responding module (00 to FF)
(data) Low latches of all channels, see Section 1.8 for the data format.

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

Examples:
Command: @01RL Response:
!01-02.000+00.000+00.000+00.000+00.000+0
0.000+00.000+00.000

Reads the low latches of module 01 and receives the data in engineering format.

Related Commands:
Section 2.34 @AACL, Section 2.35 @AACLi, Section 2.47 @AARLi

2.46. @AARLCi

Description:
This command is used to read the low alarm of a channel.

Syntax:
@AARLCi [CHKSUM](CR)
@ Delimiter character
AA Address of the module to be set (00 to FF)
RL Command to read the low alarm
Ci The channel to be clear, zero based

Response:
Valid command: **!AA(data)S[CHKSUM](CR)**
Invalid command: **?AA[CHKSUM](CR)**
! Delimiter character for a valid command
? Delimiter character for an invalid command
AA Address of the responding module (00 to FF)
(data) Low latch of a channel, see Section 1.8 for the data format.
S Alarm type:
0: alarm disable
1: momentary alarm
2: latched alarm

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

Examples:
Command: @01RLC0 Response: !01-03.0001
Reads the low alarm of channel 0 and returns the high alarm limit is -03.000 and the type is momentary.

Related Commands:
Section 2.41 @AALO(data)CiT, Section 2.38 @AADI, Section 2.36 @AADLCi

2.47. @AARLi

Description:
This command is used to read the low latch of a channel.

Syntax:
@AARLi [CHKSUM](CR)
@ Delimiter character
AA Address of the module to be set (00 to FF)
RL Command to read the low latch
i The channel to be clear, zero based

Response:
Valid command: **!AA(data)[CHKSUM](CR)**
Invalid command: **?AA[CHKSUM](CR)**
! Delimiter character for a valid command
? Delimiter character for an invalid command
AA Address of the responding module (00 to FF)
(data) High latch of a channel, see Section 1.8 for the data format.

There will be no response if the command syntax is incorrect, there is a communication error, or there is no module with the specified address.

Examples:
Command: @01RL0 Response: !01-02.000
Reads the low latch of channel 0 and returns the data in engineering format.

Related Commands:
Section 2.34 @AACL, Section 2.35 @AACLi, Section 2.45 @AARL

3.Modbus RTU Protocol

The Modbus protocol was developed by Modicon Inc., and was originally developed for Modicon controllers. Detailed information can be found at http://www.modicon.com/techpubs/toc7.html. You can also visit http://www.modbus.org to find more valuable information.

Function code	Description	Section
02 (0x02)	Read input status	3.1
04 (0x04)	Read input channels	3.2
70 (0x46)	Read/write module settings	3.3

Error Response

00	Address	1 Byte	1 to 247
01	Function code	1 Byte	Function code | 0x80
02	Egception code	1 Byte	01

If a CRC mismatch occurs, the module will not respond.

3.1. 02 (0x02) Read Input Status

This function code is used to read the wire opening status of a module. (Support type 0x7 and 0x1A only)

Request

00	Address	1 Byte	1 to 247
01	Function code	1 Byte	0x02
02 ~ 03	Starting channel	2 Bytes	0x80 to 0x87, where 0x80 corresponds to channel 0, 0x81 corresponds to channel 1, etc.
04 ~ 05	Number of input channels	2 Bytes	N, 1 to 8; (Starting channel + N)

Response

00	Address	1 Byte	1 to 247
01	Function code	1 Byte	0x02
02	Byte count	1 Byte	1
03	Data of input channels	1 Byte	A bit corresponds to a channel. When the bit is 1 denotes that the channel is either over-range or under-range. If the bit is 0 it denotes that the channel is normal.

Error Response

00	Address	1 Byte	1 to 247
01	Function code	1 Byte	0x82
02	Egception code	1 Byte	03: (the starting channel + number of input channels) is out of range, or an incorrect number of bytes were received.

3.2. 04 (0x04) Read Input Channels

This function code is used to read from contiguous analog input channels.

Request

00	Address	1 Byte	1 to 247
01	Function code	1 Byte	0x04
02 ~ 03	Starting channel	2 Bytes	0 to 7
04 ~ 05	Number of input channels (N)	2 Bytes	1 to 8

Response

00	Address	1 Byte	1 to 247
01	Function code	1 Byte	0x04
02	Byte count	1 Byte	2 x N
03 ~	Data of input channels	2 x N Bytes	Data in 2’s complement hex format or engineering format.

Error Response

00	Address	1 Byte	1 to 247
01	Function code	1 Byte	0x84
02	Egception code	1 Byte	03: (the starting channel + number of input channels) is out of range, or an incorrect number of bytes were received

3.3. 70 (0x46) Read/Write Module Settings

This function code is used to read the settings of the module or change the settings of the module. The following sub-function codes are supported.

Sub-function Code	Description	Section
00 (0x00)	Reads the module name	3.3.1
07 (0x07)	Reads the type code	3.3.2
08 (0x08)	Sets the type code	3.3.3
32 (0x20)	Reads the firmware version	3.3.4
37 (0x25)	Reads the channel enabled/disabled status	3.3.5
38 (0x26)	Sets the channel to enabled/disabled	3.3.6
41 (0x29)	Reads the miscellaneous settings	3.3.7
42 (0x2A)	Writes the miscellaneous settings	3.3.8
43 (0x2B)	Reads the CJC offset	3.3.9
44 (0x2C)	Writes the CJC offset	3.3.10
45 (0x2D)	Reads the CJC enabled/disabled status	3.3.11
46 (0x2E)	Enables/disable CJC	3.3.12
47 (0x2F)	Reads the CJC update setting	3.3.13
48 (0x30)	Sets the CJC update setting	3.3.14

If the module does not support the sub-function code specified in the message, then it will respond as follows.

Error Response

00	Address	1 Byte	1 to 247
01	Function code	1 Byte	0xC6
02	Egception code	1 Byte	02: invalid sub-function code

3.3.1 Sub-function 00 (0x00) Read module name

This sub-function code is used to read the name of a module.

Request

00	Address	1 Byte	1 to 247
01	Function code	1 Byte	0x46
02	Sub-function code	1 Byte	0x00

Response

00	Address	1 Byte	1 to 247
01	Function code	1 Byte	0x46
02	Sub-function code	1 Byte	0x00
03 ~ 06	Module name	4 Bytes	0x5A 0x70 0x17 0x00

Error Response

00	Address	1 Byte	1 to 247
01	Function code	1 Byte	0xC6
02	Egception code	1 Byte	03: an incorrect number of bytes were received

3.3.2 Sub-function 07 (0x07) Read type code

This sub-function code is used to read the type code information of a module.

Request

00	Address	1 Byte	1 to 247
01	Function code	1 Byte	0x46
02	Sub-function code	1 Byte	0x07
03	Reserved	1 Bytes	0x00
04	Channel	1 Byte	0x00 ~ 0x07

Response

00	Address	1 Byte	1 to 247
01	Function code	1 Byte	0x46
02	Sub-function code	1 Byte	0x07
03	Type code	1 Byte	Type code, see Section 1.8 for details.

Error Response

00	Address	1 Byte	1 to 247
01	Function code	1 Byte	0xC6
02	Egception code	1 Byte	03: the reserved bytes should be filled with zero, the channel is out of range for ZB-2018 series, or an incorrect number of bytes were received

3.3.3 Sub-function 08 (0x08) Set type code

This sub-function code is used to set the type code of a module.

Request

00	Address	1 Byte	1 to 247
01	Function code	1 Byte	0x46
02	Sub-function code	1 Byte	0x08
03	Reserved	1 Byte	0x00
04	Channel	1 Byte	0x00 ~ 0x07
05	Type code	1 Byte	Type code, see Section 1.8 for details.

Response

00	Address	1 Byte	1 to 247
01	Function code	1 Byte	0x46
02	Sub-function code	1 Byte	0x08
03	Type code	1 Byte	0: OK others: error

Error Response

00	Address	1 Byte	1 to 247
01	Function code	1 Byte	0xC6
02	Egception code	1 Byte	03: the type code is out of range, the channel is out of range for ZB-2018 series, reserved bytes should be filled with zero, or an incorrect number of bytes were received

3.3.4 Sub-function 32 (0x20) Read firmware version

This sub-function code is used to read the firmware version information of a module.

Request

00	Address	1 Byte	1 to 247
01	Function code	1 Byte	0x46
02	Sub-function code	1 Byte	0x20

Response

00	Address	1 Byte	1 to 247
01	Function code	1 Byte	0x46
02	Sub-function code	1 Byte	0x20
03	Major version	1 Byte	0x00 ~ 0xFF
04	Minor version	1 Byte	0x00 ~ 0xFF
05	Reserved	1 Byte	0x00
06	Build version	1 Byte	0x00 ~ 0xFF

Error Response

00	Address	1 Byte	1 to 247
01	Function code	1 Byte	0xC6
02	Egception code	1 Byte	03: an incorrect number of bytes were received

3.3.5 Sub-function 37 (0x25) Read channel enabled/disabled status

This sub-function code is used to read the enabled/disabled status of each channel in a module.

Request

00	Address	1 Byte	1 to 247
01	Function code	1 Byte	0x46
02	Sub-function code	1 Byte	0x25

Response

00	Address	1 Byte	1 to 247
01	Function code	1 Byte	0x46
02	Sub-function code	1 Byte	0x25
03	Enabled/disabled status	1 Byte	0x00 ~ 0xFF, the enabled/disabled status of each channel, where bit 0 corresponds to channel 0, bit 1 corresponds to channel 1, etc. When the bit is 1, it denotes that the channel is enabled and 0 denotes that the channel is disabled.

Error Response

00	Address	1 Byte	1 to 247
01	Function code	1 Byte	0xC6
02	Egception code	1 Byte	03: an incorrect number of bytes were received

3.3.6 Sub-function 38 (0x26) Set channel enable/disable

This sub-function code is used to specify the channels to be enabled in a module.

Request

00	Address	1 Byte	1 to 247
01	Function code	1 Byte	0x46
02	Sub-function code	1 Byte	0x26
03	Enable/disable setting	1 Byte	0x00 ~ 0xFF, the enabled/disabled settings for each channel, where bit 0 corresponds to channel 0, bit 1 corresponds to channel 1, etc. When the bit is 1, it denotes that the channel is enabled and 0 denotes that the channel is disabled.

Response

00	Address	1 Byte	1 to 247
01	Function code	1 Byte	0x46
02	Sub function code	1 Byte	0x26
03	Enable/disable setting	1 Byte	0: OK others: error.

Error Response

00	Address	1 Byte	1 to 247
01	Function code	1 Byte	0xC6
02	Egception code	1 Byte	03: the enabled/disabled settings are out of range, incorrect number of bytes were received

3.3.7 Sub-function 41 (0x29) Read miscellaneous settings

This sub-function code is used to read the miscellaneous settings of a module.

Request

00	Address	1 Byte	1 to 247
01	Function code	1 Byte	0x46
02	Sub-function code	1 Byte	0x29

Response

00	Address	1 Byte	1 to 247
01	Function code	1 Byte	0x46
02	Sub-function code	1 Byte	0x29
03	Miscellaneous settings	1 Byte	Data format, see Section 1.8 for details.

Note: The reserved fields are filled with zeros.

Error Response

00	Address	1 Byte	1 to 247
01	Function code	1 Byte	0xC6
02	Egception code	1 Byte	03: an incorrect number of bytes were received

3.3.8 Sub-function 42 (0x2A) Write miscellaneous settings

This sub-function code is used to set the miscellaneous settings of a module.

Request

00	Address	1 Byte	1 to 247
01	Function code	1 Byte	0x46
02	Sub-function code	1 Byte	0x2A
03	Miscellaneous settings	1 Byte	Data format, see Section 1.8 for details.

Note: **The reserved fields are filled with zeros.**

Response

00	Address	1 Byte	1 to 247
01	Function code	1 Byte	0x46
02	Sub-function code	1 Byte	0x2A
03	Miscellaneous settings	1 Byte	0: OK others: error

Error Response

00	Address	1 Byte	1 to 247
01	Function code	1 Byte	0xC6
02	Egception code	1 Byte	03: the reserved bits should be filled with zero, or an incorrect number of bytes were received

3.3.9 Sub-function 43 (0x2B) Read the CJC offset

This sub-function code is used to read the CJC offset of a module.

Request

00	Address	1 Byte	1 to 247
01	Function code	1 Byte	0x46
02	Sub-function code	1 Byte	0x2B
03	Reserved	1 Byte	0x00

Note: **The reserved fields are filled with zeros.**

Response

00	Address	1 Byte	1 to 247
01	Function code	1 Byte	0x46
02	Sub-function code	1 Byte	0x2B
03	CJC offset	1 Byte	This is a 2's complement hex CJC offset value in 0.01℃ increments.

Error Response

00	Address	1 Byte	1 to 247
01	Function code	1 Byte	0xC6
02	Egception code	1 Byte	03: the reserved bits should be filled with zero, or an incorrect number of bytes were received

3.3.10 Sub-function 44 (0x2C) Write the CJC offset

This sub-function code is used to set the CJC offset of a module.

Request

00	Address	1 Byte	1 to 247
01	Function code	1 Byte	0x46
02	Sub-function code	1 Byte	0x2C
03	Reserved	1 Byte	0x00
04	CJC offset	2 Bytes	This is a 2's complement hex CJC offset value in 0.01℃ increments, the absolute value should be less than or equal to 0x1000

Note: **The reserved fields are filled with zeros.**

Response

00	Address	1 Byte	1 to 247
01	Function code	1 Byte	0x46
02	Sub-function code	1 Byte	0x2C
03	Result	1 Byte	0: OK others: error

Error Response

00	Address	1 Byte	1 to 247
01	Function code	1 Byte	0xC6
02	Egception code	1 Byte	03: the reserved bits should be filled with zero, or an incorrect number of bytes were received

3.3.11 Sub-function 45 (0x2D) Read the CJC enabled/disabled status

This sub-function code is used to read the CJC enabled/disabled status of a module.

Request

00	Address	1 Byte	1 to 247
01	Function code	1 Byte	0x46
02	Sub-function code	1 Byte	0x2D
03	Reserved	1 Byte	0x00

Note: **The reserved fields are filled with zeros.**

Response

00	Address	1 Byte	1 to 247
01	Function code	1 Byte	0x46
02	Sub-function code	1 Byte	0x2D
03	CJC enabled/disabled status	1 Byte	0: CJC disabled 1: CJC enabled

Error Response

00	Address	1 Byte	1 to 247
01	Function code	1 Byte	0xC6
02	Egception code	1 Byte	03: the reserved bits should be filled with zero, or an incorrect number of bytes were received

3.3.12 Sub-function 46 (0x2E) Enable/disable CJC

This sub-function code is used to enable/disable the CJC of a module.

Request

00	Address	1 Byte	1 to 247
01	Function code	1 Byte	0x46
02	Sub-function code	1 Byte	0x2E
03	Reserved	1 Byte	0x00
04	Enable/disable CJC	1 Byte	0: disable CJC 1: enable CJC

Note: **The reserved fields are filled with zeros.**

Response

00	Address	1 Byte	1 to 247
01	Function code	1 Byte	0x46
02	Sub-function code	1 Byte	0x2E
03	Result	1 Byte	0: OK others: error

Error Response

00	Address	1 Byte	1 to 247
01	Function code	1 Byte	0xC6
02	Egception code	1 Byte	03: the reserved bits should be filled with zero, or an incorrect number of bytes were received

3.3.13 Sub-function 47 (0x2F) Read the CJC update setting

This sub-function code is used to read the CJC update setting of a module.

Request

00	Address	1 Byte	1 to 247
01	Function code	1 Byte	0x46
02	Sub-function code	1 Byte	0x2F

Note: **The reserved fields are filled with zeros.**

Response

00	Address	1 Byte	1 to 247
01	Function code	1 Byte	0x46
02	Sub-function code	1 Byte	0x2F
03	CJC temperature update setting	1 Byte	0: CJC temperature update has been stopped 1: CJC temperature update has been started 2: Update CJC temperature once only

Error Response

00	Address	1 Byte	1 to 247
01	Function code	1 Byte	0xC6
02	Egception code	1 Byte	03: the reserved bits should be filled with zero, or an incorrect number of bytes were received

3.3.14 Sub-function 48 (0x30) Write the CJC update settings

This sub-function code is used to set the CJC update settings of a module.

Request

00	Address	1 Byte	1 to 247
01	Function code	1 Byte	0x46
02	Sub-function code	1 Byte	0x30
03	CJC temperature update setting	1 Byte	0: stop CJC temperature update 1: start CJC temperature update 2: Update CJC temperature once only, after the command is received

Note: **The reserved fields are filled with zeros.**

Response

00	Address	1 Byte	1 to 247
01	Function code	1 Byte	0x46
02	Sub-function code	1 Byte	0x30
03	Result	1 Byte	0: OK others: error

Error Response

00	Address	1 Byte	1 to 247
01	Function code	1 Byte	0xC6
02	Egception code	1 Byte	03: the reserved bits should be filled with zero, or an incorrect number of bytes were received

3.4. Address Mappings

The address mappings are as follows.

Address	Description	Attribute
00259	Filter settings, 0: 60Hz rejection, 1: 50Hz rejection	R/W
00260	Modbus host watchdog mode 0: same as I-7000 1: the AO and DO commands can be used to clear the host watchdog timeout status	R/W
00261	Enable/disable the host watchdog 0: disable 1: enable	R/W
00268	Enable/disable CJC 0: disable 1: enable	R/W
00269	Modbus data format 0: hex 1: engineering	R/W
00270	Host watch dog timeout status, write 1 to clear the host watch dog timeout status	W
00272	Write 1 to load the factory calibration parameters	W
00273	Reset status 0: not the first read after being powered on 1: first read after being powered on	R
00279	CJC Status 0: good 1: bad	R
00280	High latch of channel 0 to 7, write 1 to clear	W
00281	Low latch of channel 0 to 7, write 1 to clear	W

00513 ~ 00520	High latch of channel 0 to 7, write 1 to clear	W
00545 ~ 00552	Low latch of channel 0 to 7, write 1 to clear	W
00577 ~ 00584	Enable/disable the hugh alarm of channel 0 to 7 1: enable 0: disable	R/W
00609 ~ 00616	Enable/disable the low alarm of channel 0 to 7 1: enable 0: disable	R/W
00641 ~ 00648	High alarm mode of channel 0 to 7 1: latch 0: momentary	R/W
00673 ~ 00680	Low alarm mode of channel 0 to 7 1: latch 0: momentary	R/W
00705 ~ 00712	High alarm status of channel 0 to 7	R/W
00737 ~ 00744	Low alarm status of channel 0 to 7	R/W
10129 ~ 10136	Under range status of channel 0 to 7 (support type 0x7 and 0xE ~ 0x1A only)	R
30001 ~ 30008	Analog input value of channel 0 to 7	R
30129	CJC temperature	R
30513 ~ 30520	High latch value	R
30545 ~ 30552	Low latch value	R
40257 ~ 40264	Type code of channel 0 to 7	R/W
40353 ~ 40360	CJC offset of a channel	R/W
40481	Firmware version (low word)	R

40482	Firmware version (high word)	R
40483	Module name (low word)	R
40484	Module name (high word)	R
40485	Module address, valid range: 0x1 ~ 0x1F	R
40486	Bits 5:0 Baud Rate, 0x0A Bits 7:6 Reserved	R
40489	Host watchdog timeout value, 0 ~ 255, in 0.1s	R/W
40490	Channel enabled/disabled	R/W
40491	CJC offset of channel 0 to 7	R/W
40492	Host watchdog timeout count, write 0 to clear	R/W
40493	CJC update setting 0: stop CJC temperature update 1: start CJC temperature update 2: update CJC temperature once only, after set the command.	R/W
40577 ~ 40584	High alarm value	R/W
40609 ~ 40616	Low alarm value	R/W

Notes:
1. The command for loading factory calibration parameters takes about 3 seconds to be processed. The next command should be sent after 3 seconds.

3.5. Engineering Data Format Table

The engineering data format for the Modbus protocol is supported, and the table is as follows.

Type Code	Analog Input Type	-F.S.	+F.S.
00	-15 to +15 mV	-15000	15000
01	-50 to +50 mV	-5000	5000
02	-100 to +100 mV	-10000	10000
03	-500 to +500 mV	-5000	5000
04	-1 to +1 V	-10000	10000
05	-2.5 to +2.5 V	-25000	25000
06	-20 to +20 mA	-20000	20000
07	+4 to +20 mA	4000	20000
0E	Type J Termocouple	-2100	7600
0F	Type K Termocouple	-2700	13720
10	Type T Termocouple	-2700	4000
11	Type E Termocouple	-2700	10000
12	Type R Termocouple	0	17680
13	Type S Termocouple	0	17680
14	Type B Termocouple	0	18200

15	Type N Termocouple	-2700	13000
16	Type C Termocouple	0	23200
17	Type L Termocouple	-2000	8000
18	Type M Termocouple	-20000	10000
19	Type $L_{DIN43710}$ Termocouple	-2000	9000
1A	0 to +20 mA	0	20000

The under range value is –32768 and the over range value is +32767.
For the hex data format, please refer to Section 1.8 for details.

4. Troubleshooting

If you are having difficulty using the ZB-2018 series module, here are some suggestions that may help. If you cannot find the answers you need in these guides, contact ICP DAS Product Support. Contact information is located in Section 1.10.

4.1. Communicating with the module

If you attempt to communicate with the module and receive no response, first check the following:

- Ensure that the supplied power is within the range of +10 to +30 V DC. If the supplied power is OK, then the power LED should be on.
- When the module receives a command, the power LED is set to “off”. The power LED is shown as “on” after the module responds. This method can be used to check whether the module has received a command sent from the host.
- If possible, use another device that is known to be functional to check whether the host can communicate with the device through the same ZigBee network.
- If the host is a PC installed with a Windows operating system, then execute the DCON Utility to determine whether the module can be found. The DCON Utility can be downloaded from the ICP DAS website http://www.icpdas.com. The DCON Utility documentation can be found in the ”G**etting Started For I-7000 Series Modules**” manual.
- Set the module to “INIT mode” and communicate with the module using the following settings: address 00 and DCON protocol. See Section 1.7 for details.

4.2. Reading Data

If the data read from the input channel is not correct, first check the following:

- Ensure that the type code and data format settings are correct. The type code is set by using the $AA7CiRrr command, see Section 2.11 for details. The data format is set by using the %AANNTTCCFF command. For the Modbus RTU protocol, the type code is set by using sub-function 08h of the function 46h.
- If the voltage read by the module is incorrect, then it may be because the calibration parameters stores in the non-voltage memory are corrupted. You can calibrate the module by yourself. Be sure to read Section 1.9 for details before doing any calibration. Use the $AAS1 command to reload the factory calibration parameters, see Section 2.21 for details.

5. Appendix

5.1. Dual Watchdog Operation

Dual Watchdog = Module Watchdog + Host Watchdog

The Module Watchdog is a hardware reset circuit that monitors the operating status of the module. While working in harsh or noisy environments, the module may be shut down by external signals. The circuit allows the module to work continuously without disruption.

The Host Watchdog is a software function that monitors the operating status of the host. Its purpose is to prevent problems due to network/communication errors or host malfunctions. When a host watchdog timeout occurs, the module will reset all outputs to a safe state in order to prevent any erroneous operations of the controlled target.

ZB-2000 series modules include an internal Dual Watchdog, making the control system more reliable and stable.

For more information regarding the Dual Watchdog, please refer to Chapter 5 of the “Getting Started For ZB-2018 series Modules” manual that can be downloaded from the ICP DAS website http://www.icpdas.com.